La Jerarquía Polinomial

IIC3242

Motivación: Soluciones exactas

Al estudiar la clase NP consideramos preguntas "existenciales":

- ¿Existe una asignación de verdad que satisface a una fórmula proposicional?
- ¿Existe una solución entera para un sistema de ecuaciones lineales enteras?
- ¿Existe un camino Hamiltoniano en un grafo?

¿Cambia la complejidad de estos problemas si hacemos preguntas "exactas"?

Motivación: Soluciones exactas

Al estudiar la clase NP consideramos preguntas "existenciales":

- ¿Existe una asignación de verdad que satisface a una fórmula proposicional?
- ¿Existe una solución entera para un sistema de ecuaciones lineales enteras?
- ¿Existe un camino Hamiltoniano en un grafo?

¿Cambia la complejidad de estos problemas si hacemos preguntas "exactas"?

- ¿Existe exactamente una asignación de verdad que satisface a una fórmula proposicional?

Motivación: Soluciones exactas

Al estudiar la clase NP consideramos preguntas "existenciales":

- ¿Existe una asignación de verdad que satisface a una fórmula proposicional?
- ¿Existe una solución entera para un sistema de ecuaciones lineales enteras?
- ¿Existe un camino Hamiltoniano en un grafo?

¿Cambia la complejidad de estos problemas si hacemos preguntas "exactas"?

- ¿Existe exactamente una asignación de verdad que satisface a una fórmula proposicional?
- ¿Cuál es la complejidad de este problema? ¿Está en NP?

Soluciones exactas: La clase DP

Para responder a las preguntas anteriores necesitamos definir una nueva clase de complejidad.

Idea detrás de esta clase: Noción de solución exacta se reduce a hacer dos consultas.

- ¿Es cierto que existe una asignación de verdad que satisface la fórmula proposicional φ?
- ¿Es cierto que no existen al menos dos asignaciones de verdad que satisfacen φ?

Soluciones exactas: La clase DP

Para responder a las preguntas anteriores necesitamos definir una nueva clase de complejidad.

Idea detrás de esta clase: Noción de solución exacta se reduce a hacer dos consultas.

- ¿Es cierto que existe una asignación de verdad que satisface la fórmula proposicional φ?
- ¿Es cierto que no existen al menos dos asignaciones de verdad que satisfacen φ?

Definición (Clase de complejidad DP)

$L \in DP$ si y sólo si existen lenguajes $L_1 \in NP$ y $L_2 \in co\text{-}NP$ tales que $L = L_1 \cap L_2$.

unique-SAT y la clase DP

Una definición formal del primer problema que queremos estudiar:

$$\text{unique-SAT} = \{\varphi \mid \varphi \text{ es una fórmula proposicional tal que existe exactamente una asignación de verdad que satisface } \varphi\}$$

Ejercicio

Demuestre que unique-SAT $\in$ DP.

unique-SAT y la clase DP

Una definición formal del primer problema que queremos estudiar:

$$\text{unique-SAT} = \{\varphi \mid \varphi \text{ es una fórmula proposicional tal que existe exactamente una asignación de verdad que satisface } \varphi\}$$

Ejercicio

Demuestre que unique-SAT $\in$ DP.

¿Es unique-SAT completo para DP?

unique-SAT y la clase DP

Una definición formal del primer problema que queremos estudiar:

$$\text{unique-SAT} = \{\varphi \mid \varphi \text{ es una fórmula proposicional tal que existe exactamente una asignación de verdad que satisface } \varphi\}$$

Ejercicio

Demuestre que unique-SAT $\in$ DP.

¿Es unique-SAT completo para DP?

- Este es un problema abierto

unique-SAT y la clase DP

Una definición formal del primer problema que queremos estudiar:

$$\text{unique-SAT} = \{\varphi \mid \varphi \text{ es una fórmula proposicional tal que existe exactamente una asignación de verdad que satisface } \varphi\}$$

Ejercicio

Demuestre que unique-SAT $\in$ DP.

¿Es unique-SAT completo para DP?

- Este es un problema abierto
- ¿Existen problemas completos para esta clase?

Un primer problema completo para DP

Sea 3-CNF-SAT-UNSAT el siguiente lenguaje:

$$\text{3-CNF-SAT-UNSAT} = \{(\varphi, \psi) \mid \varphi \text{ y } \psi \text{ son conjunciones de 3-clausulas, } \varphi \text{ es satisfacible y } \psi \text{ no es satisfacible}\}$$

Un primer problema completo para DP

Sea 3-CNF-SAT-UNSAT el siguiente lenguaje:

$$\text{3-CNF-SAT-UNSAT} = \{(\varphi, \psi) \mid \varphi \text{ y } \psi \text{ son conjunciones de 3-clausulas, } \varphi \text{ es satisfacible y } \psi \text{ no es satisfacible}\}$$

Teorema

3-CNF-SAT-UNSAT es DP-completo.

Un primer problema completo para DP

Sea 3-CNF-SAT-UNSAT el siguiente lenguaje:

$$\text{3-CNF-SAT-UNSAT} = \{(\varphi, \psi) \mid \varphi \text{ y } \psi \text{ son conjunciones de 3-clausulas, } \varphi \text{ es satisfacible y } \psi \text{ no es satisfacible}\}$$

Teorema

3-CNF-SAT-UNSAT es DP-completo.

Ejercicio

Demuestre el teorema.

Un problema natural que representan a DP

Notación

$$\textit{clique-number}(G) \quad = \quad \text{máx}\{k \mid G \textit{ tiene un clique de tamaño } k\}$$

Usamos este número para definir dos problemas en grafos:

$$\begin{aligned} \text{CLIQUE} &= \{(G,k) \mid \text{clique-number}(G) \geq k\} \\ \text{exact-CLIQUE} &= \{(G,k) \mid \text{clique-number}(G) = k\} \end{aligned}$$

Un problema natural que representan a DP

Notación

$$\textit{clique-number}(G) \quad = \quad \text{máx}\{k \mid G \textit{ tiene un clique de tamaño } k\}$$

Usamos este número para definir dos problemas en grafos:

$$\begin{aligned} \text{CLIQUE} &= \{(G,k) \mid \text{clique-number}(G) \geq k\} \\ \text{exact-CLIQUE} &= \{(G,k) \mid \text{clique-number}(G) = k\} \end{aligned}$$

¿Cuál es la complejidad de CLIQUE?

- ¿Son distintas las complejidades de CLIQUE y exact-CLIQUE?

Un problema natural que representan a DP

Teorema

exact-CLIQUE es DP-completo.

Un problema natural que representan a DP

Teorema

exact-CLIQUE es DP-completo.

Demostración: Primero tenemos que demostrar que exact-CLIQUE $\in$ DP

- ¿Cómo se demuestra esto?

Después tenemos que demostrar que exact-CLIQUE es DP-hard.

Un problema natural que representan a DP

Teorema

exact-CLIQUE es DP-completo.

Demostración: Primero tenemos que demostrar que exact-CLIQUE $\in$ DP

- ¿Cómo se demuestra esto?

Después tenemos que demostrar que exact-CLIQUE es DP-hard.

- Vamos a reducir 3-CNF-SAT-UNSAT a exact-CLIQUE

exact-CLIQUE es DP-hard: Demostración

Vamos a utilizar una reducción usual de 3-CNF-SAT a CLIQUE.

exact-CLIQUE es DP-hard: Demostración

Vamos a utilizar una reducción usual de 3-CNF-SAT a CLIQUE.

- Recordemos esta reducción a través del ejemplo:

$$\beta \quad = \quad (p \vee q \vee r) \wedge (\neg p \vee \neg q \vee s) \wedge (p \vee \neg q \vee \neg s)$$

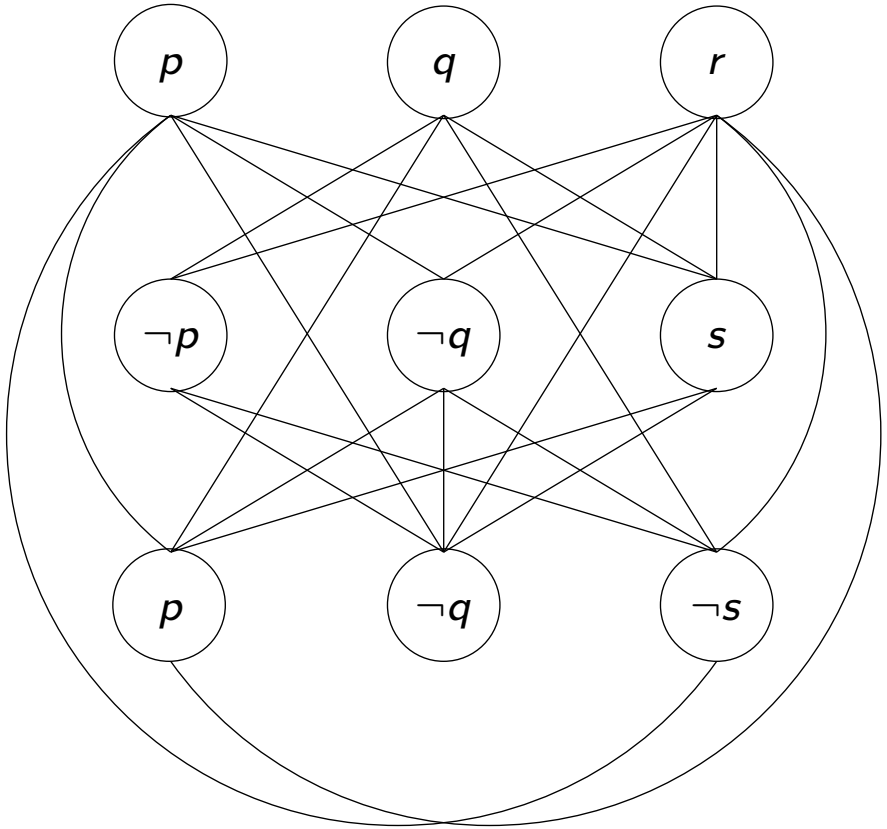
exact-CLIQUE es DP-hard: Demostración

Vamos a utilizar una reducción usual de 3-CNF-SAT a CLIQUE.

- Recordemos esta reducción a través del ejemplo:

$$\beta \quad = \quad (p \vee q \vee r) \wedge (\neg p \vee \neg q \vee s) \wedge (p \vee \neg q \vee \neg s)$$

Grafo G_β:



exact-CLIQUE es DP-hard: Demostración

Tenemos que: $\beta \in$ 3-CNF-SAT si y sólo si $(G_\beta, 3) \in$ CLIQUE

exact-CLIQUE es DP-hard: Demostración

Tenemos que: $\beta \in$ 3-CNF-SAT si y sólo si $(G_\beta, 3) \in$ CLIQUE

En general: Si $\varphi = C_1 \wedge \cdots \wedge C_n$, donde cada C_i es una clausula, entonces:

$$\varphi \in \text{3-CNF-SAT} \quad \text{si y sólo si} \quad (G_\varphi, n) \in \text{CLIQUE}$$

exact-CLIQUE es DP-hard: Demostración

Tenemos que: $\beta \in$ 3-CNF-SAT si y sólo si $(G_\beta, 3) \in$ CLIQUE

En general: Si $\varphi = C_1 \wedge \cdots \wedge C_n$, donde cada C_i es una clausula, entonces:

$$\varphi \in \text{3-CNF-SAT} \quad \text{si y sólo si} \quad (G_\varphi, n) \in \text{CLIQUE}$$

Vamos a utilizar esta reducción en nuestra demostración.

- Pero antes: Necesitamos algunas operaciones sobre grafos

exact-CLIQUE es DP-hard: Demostración

Dados: Grafos $G_1 = (N_1, A_1)$ y $G_2 = (N_2, A_2)$

exact-CLIQUE es DP-hard: Demostración

Dados: Grafos $G_1 = (N_1, A_1)$ y $G_2 = (N_2, A_2)$

Notación

$G_1 \uplus G_2$: *Unión disjunta de* G_1 *y* G_2

- *Suponiendo que* $N_1 \cap N_2 = \emptyset$: $G_1 \uplus G_2 = (N_1 \cup N_2, A_1 \cup A_2)$

exact-CLIQUE es DP-hard: Demostración

Dados: Grafos $G_1 = (N_1, A_1)$ y $G_2 = (N_2, A_2)$

Notación

$G_1 \uplus G_2$: *Unión disjunta de* G_1 *y* G_2

- *Suponiendo que* $N_1 \cap N_2 = \emptyset$: $G_1 \uplus G_2 = (N_1 \cup N_2, A_1 \cup A_2)$

Si $N_1 \cap N_2 \neq \emptyset$: Los nodos de G_2 pueden ser renombrados para poder ejecutar la operación $\uplus$

exact-CLIQUE es DP-hard: Demostración

Notación

$G_1 \times G_2 = (N_1 \times N_2, A)$, *donde:*

$$\begin{aligned} A \;=\; \{((a_1,a_2),(b_1,b_2)) \mid\; & (a_1,b_1) \in A_1 \text{ y } (a_2,b_2) \in A_2, \text{ ó} \\ & a_1 = b_1 \text{ y } (a_2,b_2) \in A_2, \text{ ó} \\ & (a_1,b_1) \in A_1 \text{ y } a_2 = b_2\} \end{aligned}$$

exact-CLIQUE es DP-hard: Demostración

Notación

$G_1 \times G_2 = (N_1 \times N_2, A)$, *donde:*

$$A = \{((a_1, a_2), (b_1, b_2)) \mid (a_1, b_1) \in A_1 \text{ y } (a_2, b_2) \in A_2, \text{ ó } a_1 = b_1 \text{ y } (a_2, b_2) \in A_2, \text{ ó } (a_1, b_1) \in A_1 \text{ y } a_2 = b_2\}$$

Ejercicio

Si G_1 y G_2 tienen cliques con n_1 y n_2 elementos, respectivamente, ¿qué puede decir sobre los cliques en $G_1 \times G_2$?

exact-CLIQUE es DP-hard: Demostración

Tenemos los ingredientes para reducir de 3-CNF-SAT-UNSAT a exact-CLIQUE.

Dado (φ, ψ), vamos a construir $(G_{(\varphi,\psi)}, k_{(\varphi,\psi)})$ tal que:

$$(\varphi, \psi) \in \text{3-CNF-SAT-UNSAT}$$
$$\text{si y sólo si}$$
$$(G_{(\varphi,\psi)}, k_{(\varphi,\psi)}) \in \text{exact-CLIQUE}$$

exact-CLIQUE es DP-hard: Demostración

Tenemos los ingredientes para reducir de 3-CNF-SAT-UNSAT a exact-CLIQUE.

Dado (φ, ψ), vamos a construir $(G_{(\varphi,\psi)}, k_{(\varphi,\psi)})$ tal que:

$$(\varphi, \psi) \in \text{3-CNF-SAT-UNSAT}$$
$$\text{si y sólo si}$$
$$(G_{(\varphi,\psi)}, k_{(\varphi,\psi)}) \in \text{exact-CLIQUE}$$

Suponemos que φ y ψ están formadas por m y n clausulas, respectivamente, donde $m \geq 2$, $n \geq 2$ y $m \neq n$.

exact-CLIQUE es DP-hard: Demostración

Tenemos los ingredientes para reducir de 3-CNF-SAT-UNSAT a exact-CLIQUE.

Dado (φ, ψ), vamos a construir $(G_{(\varphi,\psi)}, k_{(\varphi,\psi)})$ tal que:

$$(\varphi, \psi) \in \text{3-CNF-SAT-UNSAT}$$
$$\text{si y sólo si}$$
$$(G_{(\varphi,\psi)}, k_{(\varphi,\psi)}) \in \text{exact-CLIQUE}$$

Suponemos que φ y ψ están formadas por m y n clausulas, respectivamente, donde $m \geq 2$, $n \geq 2$ y $m \neq n$.

- ¿Por qué podemos suponer que $m \geq 2$, $n \geq 2$ y $m \neq n$?

exact-CLIQUE es DP-hard: Demostración

Sea:

$$\begin{aligned} G_{(\varphi,\psi)} &= (G_\varphi \uplus K_{m-1}) \times (G_\psi \uplus K_{n-1}) \\ k_{(\varphi,\psi)} &= m \cdot (n-1) \end{aligned}$$

exact-CLIQUE es DP-hard: Demostración

Sea:

$$\begin{aligned} G_{(\varphi,\psi)} &= (G_\varphi \uplus K_{m-1}) \times (G_\psi \uplus K_{n-1}) \\ k_{(\varphi,\psi)} &= m \cdot (n-1) \end{aligned}$$

Tenemos que:

φ	ψ	Tamaño del mayor clique en $G_{(\varphi,\psi)}$
CNF-SAT	CNF-SAT	$m \cdot n$
CNF-SAT	$\overline{\text{CNF-SAT}}$	$m \cdot (n-1)$
$\overline{\text{CNF-SAT}}$	CNF-SAT	$(m-1) \cdot n$
$\overline{\text{CNF-SAT}}$	$\overline{\text{CNF-SAT}}$	$(m-1) \cdot (n-1)$

exact-CLIQUE es DP-hard: Demostración

Dado que $m \neq n$: $m \cdot (n-1) \neq (m-1) \cdot n$

Por lo tanto, dado que $(m-1) \cdot (n-1) < m \cdot (n-1) < m \cdot n$, concluimos que:

$$(\varphi, \psi) \in \text{3-CNF-SAT-UNSAT}$$
$$\text{si y sólo si}$$
$$(G_{(\varphi,\psi)}, k_{(\varphi,\psi)}) \in \text{exact-CLIQUE}$$

exact-CLIQUE es DP-hard: Demostración

Dado que $m \neq n$: $m \cdot (n-1) \neq (m-1) \cdot n$

Por lo tanto, dado que $(m-1) \cdot (n-1) < m \cdot (n-1) < m \cdot n$, concluimos que:

$$(\varphi, \psi) \in \text{3-CNF-SAT-UNSAT}$$
$$\text{si y sólo si}$$
$$(G_{(\varphi,\psi)}, k_{(\varphi,\psi)}) \in \text{exact-CLIQUE}$$

Para concluir la demostración sólo nos falta demostrar que la reducción puede ser implementada en espacio logarítmico.

- ¿Cómo se demuestra esto?

La relación entre NP y DP

Es fácil ver que NP $\subseteq$ DP y co-NP $\subseteq$ DP.

- ¿Cómo se demuestra esto?

¿Cuál es la relación entre NP y DP?

- ¿Es NP $\neq$ DP?

La relación entre NP y DP

Es fácil ver que NP $\subseteq$ DP y co-NP $\subseteq$ DP.

- ¿Cómo se demuestra esto?

¿Cuál es la relación entre NP y DP?

- ¿Es NP $\neq$ DP?
- Este es un problema abierto, y hay una buena explicación de por qué

La relación entre NP y DP

Teorema

NP = DP si y sólo si NP = co-NP.

La relación entre NP y DP

Teorema

NP = DP si y sólo si NP = co-NP.

Demostración: ($\Leftarrow$) Suponga que NP = co-NP.

Si $L \in$ DP: $L = L_1 \cap L_2$, con $L_1 \in$ NP y $L_2 \in$ co-NP.

- Por hipótesis: $L_2 \in$ NP

Por lo tanto: $L = L_1 \cap L_2$, con $L_1, L_2 \in$ NP.

Pero NP es cerrado bajo intersección.

- Concluimos que $L \in$ NP

La relación entre NP y DP

($\Rightarrow$) Suponga que NP $=$ DP.

Como co-NP $\subseteq$ DP: co-NP $\subseteq$ NP

Por lo que también tenemos que NP $\subseteq$ co-NP:

La relación entre NP y DP

($\Rightarrow$) Suponga que NP $=$ DP.

Como co-NP $\subseteq$ DP: co-NP $\subseteq$ NP

Por lo que también tenemos que NP $\subseteq$ co-NP:

$L \in$ NP

La relación entre NP y DP

($\Rightarrow$) Suponga que NP = DP.

Como co-NP $\subseteq$ DP: co-NP $\subseteq$ NP

Por lo que también tenemos que NP $\subseteq$ co-NP:

$$L \in \text{NP} \quad \Rightarrow \quad \overline{L} \in \text{co-NP}$$

La relación entre NP y DP

($\Rightarrow$) Suponga que NP = DP.

Como co-NP $\subseteq$ DP: co-NP $\subseteq$ NP

Por lo que también tenemos que NP $\subseteq$ co-NP:

$$\begin{aligned} L \in \text{NP} &\Rightarrow \overline{L} \in \text{co-NP} \\ &\Rightarrow \overline{L} \in \text{NP} \qquad \text{(ya que co-NP} \subseteq \text{NP)} \end{aligned}$$

La relación entre NP y DP

($\Rightarrow$) Suponga que NP = DP.

Como co-NP $\subseteq$ DP: co-NP $\subseteq$ NP

Por lo que también tenemos que NP $\subseteq$ co-NP:

$$\begin{aligned} L \in \text{NP} &\Rightarrow \overline{L} \in \text{co-NP} \\ &\Rightarrow \overline{L} \in \text{NP} \qquad \text{(ya que co-NP} \subseteq \text{NP)} \\ &\Rightarrow L \in \text{co-NP} \end{aligned}$$

Motivación: Problemas de optimización

Un tour π en un grafo G es una secuencia de arcos (a_1, a_2), ..., (a_{k-1}, a_k), (a_k, a_1) en G tal que:

- $a_i \neq a_j$ para cada $i \neq j$,
- $\{a_1, \ldots, a_k\}$ es el conjunto de nodos de G.

Notación

- *Una función de costo para un grafo $G = (N, A)$ es una función costo $: A \to \mathbb{N}$.*
- *Costo de un tour π en G:*

$$costo(\pi) \quad = \quad \left(\sum_{i=1}^{k-1} costo((a_i, a_{i+1})) \right) + costo((a_k, a_1))$$

Motivación: Problemas de optimización

El problema del agente viajero se define como:

$$\text{TSP} = \{(G, \text{costo}, k) \mid \text{existe un tour } \pi \text{ en } G \text{ tal que costo}(\pi) \leq k\}$$

¿Cuál es la complejidad de TSP?

Motivación: Problemas de optimización

El problema del agente viajero se define como:

$$\text{TSP} = \{(G, \text{costo}, k) \mid \text{existe un tour } \pi \text{ en } G \text{ tal que costo}(\pi) \leq k\}$$

¿Cuál es la complejidad de TSP?

Teorema

TSP es NP-completo.

Motivación: Problemas de optimización

TSP es un problema de decisión.

Motivación: Problemas de optimización

TSP es un problema de decisión.

- Existe una versión de optimización de TSP: Encuentre el menor k tal que $(G, \text{costo}, k) \in \text{TSP}$

Motivación: Problemas de optimización

TSP es un problema de decisión.

- Existe una versión de optimización de TSP: Encuentre el menor k tal que $(G, \text{costo}, k) \in \text{TSP}$
- El problema de optimización es al menos tan difícil como el problema de decisión

Motivación: Problemas de optimización

TSP es un problema de decisión.

- Existe una versión de optimización de TSP: Encuentre el menor k tal que $(G, \text{costo}, k) \in \text{TSP}$
- El problema de optimización es al menos tan difícil como el problema de decisión
- Si se puede resolver TSP en tiempo polinomial, ¿se puede resolver la versión de optimización en tiempo polinomial?

Motivación: Problemas de optimización

TSP es un problema de decisión.

- Existe una versión de optimización de TSP: Encuentre el menor k tal que $(G, \text{costo}, k) \in \text{TSP}$
- El problema de optimización es al menos tan difícil como el problema de decisión
- Si se puede resolver TSP en tiempo polinomial, ¿se puede resolver la versión de optimización en tiempo polinomial?
  - TSP tiene que ser utilizado un número polinomial de veces

Motivación: Problemas de optimización

Un tour es óptimo si no existe otro tour con costo menor.

- Puede haber más de un tour óptimo

Otra versión de TSP:

$$\text{unique-TSP} = \{(G, \text{costo}) \mid \text{existe un único tour óptimo para } G\}$$

Este es un problema de decisión: Parece ser más difícil que TSP.

Motivación: Problemas de optimización

Un tour es óptimo si no existe otro tour con costo menor.

- Puede haber más de un tour óptimo

Otra versión de TSP:

$$\text{unique-TSP} = \{(G, \text{costo}) \mid \text{existe un único tour óptimo para } G\}$$

Este es un problema de decisión: Parece ser más difícil que TSP.

- Si se puede resolver TSP en tiempo polinomial, ¿se puede resolver unique-TSP en tiempo polinomial?

Motivación: Problemas de optimización

Un tour es óptimo si no existe otro tour con costo menor.

- Puede haber más de un tour óptimo

Otra versión de TSP:

$$\text{unique-TSP} = \{(G, \text{costo}) \mid \text{existe un único tour óptimo para } G\}$$

Este es un problema de decisión: Parece ser más difícil que TSP.

- Si se puede resolver TSP en tiempo polinomial, ¿se puede resolver unique-TSP en tiempo polinomial?
  - Nuevamente TSP tiene que ser utilizado un número polinomial de veces

La noción de oráculo

¿Qué tienen en común los problemas anteriores?

La noción de oráculo

¿Qué tienen en común los problemas anteriores?

- Se puede resolver estos problemas utilizando problemas en NP un número polinomial de veces

La noción de oráculo

¿Qué tienen en común los problemas anteriores?

- Se puede resolver estos problemas utilizando problemas en NP un número polinomial de veces
  - Para el caso de DP: Dos veces

La noción de oráculo

¿Qué tienen en común los problemas anteriores?

- Se puede resolver estos problemas utilizando problemas en NP un número polinomial de veces
  - Para el caso de DP: Dos veces
- Si encontramos un algoritmo polinomial para un problema NP-completo, entonces estos problemas pueden ser resueltos en tiempo polinomial

La noción de oráculo

¿Qué tienen en común los problemas anteriores?

- Se puede resolver estos problemas utilizando problemas en NP un número polinomial de veces
  - Para el caso de DP: Dos veces
- Si encontramos un algoritmo polinomial para un problema NP-completo, entonces estos problemas pueden ser resueltos en tiempo polinomial
- Un problema NP-completo puede ser visto como un *oráculo* para estos problemas

La noción de oráculo

¿Qué tienen en común los problemas anteriores?

- Se puede resolver estos problemas utilizando problemas en NP un número polinomial de veces
  - Para el caso de DP: Dos veces
- Si encontramos un algoritmo polinomial para un problema NP-completo, entonces estos problemas pueden ser resueltos en tiempo polinomial
- Un problema NP-completo puede ser visto como un *oráculo* para estos problemas

Vamos a introducir la noción de MT con oráculo.

La noción de oráculo

¿Qué tienen en común los problemas anteriores?

- Se puede resolver estos problemas utilizando problemas en NP un número polinomial de veces
  - Para el caso de DP: Dos veces
- Si encontramos un algoritmo polinomial para un problema NP-completo, entonces estos problemas pueden ser resueltos en tiempo polinomial
- Un problema NP-completo puede ser visto como un *oráculo* para estos problemas

Vamos a introducir la noción de MT con oráculo.

- Vamos a mostrar que sirve para caracterizar a los problemas anteriores, y a muchos otros problemas ...

MT con oráculo

Definición

MT determinista con una cinta de trabajo y oráculo para $A \subseteq \Sigma^*$:
$M^A = (Q, \Sigma, \Gamma, q_0, \delta, F)$

- Q *es un conjunto finito de estados tal que* $q_?, q_{YES}, q_{NO} \in Q$
- Σ *es un alfabeto finito tal que* $\vdash, \mathtt{B} \notin \Sigma$
- Γ *es un alfabeto finito tal que* $\Sigma \cup \{\vdash, \mathtt{B}\} \subseteq \Gamma$
- $q_0 \in Q$ *es el estado inicial*
- $F \subseteq Q$ *es un conjunto de estados finales*
- δ *es una función parcial:*

$$\delta \quad : \quad Q \times \Gamma \times \Gamma^2 \to Q \times \{\leftarrow, \square, \rightarrow\} \times \Gamma^2 \times \{\leftarrow, \square, \rightarrow\}^2$$

La tercera cinta es la cinta de consulta

MT con oráculo: Funcionamiento

La definición anterior puede extenderse fácilmente al caso de no determinismo y/o k cintas de trabajo.

MT con oráculo: Funcionamiento

La definición anterior puede extenderse fácilmente al caso de no determinismo y/o k cintas de trabajo.

Una MT M con oráculo para A funciona como una MT tradicional excepto cuando entra al esto $q_?$:

MT con oráculo: Funcionamiento

La definición anterior puede extenderse fácilmente al caso de no determinismo y/o k cintas de trabajo.

Una MT M con oráculo para A funciona como una MT tradicional excepto cuando entra al esto $q_?$:

- Cinta de consulta: $\vdash wBB\cdots$, para $w \in \Sigma^*$

  La cabeza lectora de la cinta de consulta está en la posición 1

MT con oráculo: Funcionamiento

La definición anterior puede extenderse fácilmente al caso de no determinismo y/o k cintas de trabajo.

Una MT M con oráculo para A funciona como una MT tradicional excepto cuando entra al esto $q_?$:

- Cinta de consulta: $\vdash wBB\cdots$, para $w \in \Sigma^*$

  La cabeza lectora de la cinta de consulta está en la posición 1

- M invoca al oráculo para A, y su siguiente estado es q_{YES} o q_{NO} dependiendo de su respuesta
  - $w \in A$ si y sólo si el estado es q_{YES}

MT con oráculo: Tiempo de ejecución

El tiempo de ejecución de una MT con oráculo se define como para el caso de las MTs tradicionales.

- Una invocación al oráculo se considera como un paso

Para los ejemplos iniciales:

MT con oráculo: Tiempo de ejecución

El tiempo de ejecución de una MT con oráculo se define como para el caso de las MTs tradicionales.

- Una invocación al oráculo se considera como un paso

Para los ejemplos iniciales:

3-CNF-SAT-UNSAT : aceptado por M^{SAT} que funciona en tiempo $O(n)$

MT con oráculo: Tiempo de ejecución

El tiempo de ejecución de una MT con oráculo se define como para el caso de las MTs tradicionales.

- Una invocación al oráculo se considera como un paso

Para los ejemplos iniciales:

3-CNF-SAT-UNSAT	:	aceptado por M^{SAT} que funciona en tiempo $O(n)$
exact-CLIQUE	:	aceptado por M^{CLIQUE} que funciona en tiempo $O(n)$

MT con oráculo: Tiempo de ejecución

El tiempo de ejecución de una MT con oráculo se define como para el caso de las MTs tradicionales.

- Una invocación al oráculo se considera como un paso

Para los ejemplos iniciales:

3-CNF-SAT-UNSAT	:	aceptado por M^{SAT} que funciona en tiempo $O(n)$
exact-CLIQUE	:	aceptado por M^{CLIQUE} que funciona en tiempo $O(n)$
unique-TSP	:	aceptado por M^{TSP} que funciona en tiempo $O(n^k)$

Clases de complejidad y la noción de oráculo

Una primera clase definida en términos de MTs con oráculos:

Definición

$PTIME^A$: *Lenguajes L para los cuales existe una MT determinista M^A tal que $L = L(M^A)$ y M^A funciona en tiempo $O(n^k)$.*

Clases de complejidad y la noción de oráculo

Una primera clase definida en términos de MTs con oráculos:

Definición

$PTIME^A$: Lenguajes L para los cuales existe una MT determinista M^A tal que $L = L(M^A)$ y M^A funciona en tiempo $O(n^k)$.

Tenemos que:

Clases de complejidad y la noción de oráculo

Una primera clase definida en términos de MTs con oráculos:

Definición

$PTIME^A$: Lenguajes L para los cuales existe una MT determinista M^A tal que $L = L(M^A)$ y M^A funciona en tiempo $O(n^k)$.

Tenemos que:

- 3-CNF-SAT-UNSAT, exact-CLIQUE y unique-TSP están en $\text{PTIME}^{\text{SAT}}$

Clases de complejidad y la noción de oráculo

Una primera clase definida en términos de MTs con oráculos:

Definición

$PTIME^A$: Lenguajes L para los cuales existe una MT determinista M^A tal que $L = L(M^A)$ y M^A funciona en tiempo $O(n^k)$.

Tenemos que:

- 3-CNF-SAT-UNSAT, exact-CLIQUE y unique-TSP están en $\text{PTIME}^{\text{SAT}}$
  - También están contenidos en PTIME^A, para A problema NP-completo

Clases de complejidad y la noción de oráculo

Una primera clase definida en términos de MTs con oráculos:

Definición

$PTIME^A$: Lenguajes L para los cuales existe una MT determinista M^A tal que $L = L(M^A)$ y M^A funciona en tiempo $O(n^k)$.

Tenemos que:

- 3-CNF-SAT-UNSAT, exact-CLIQUE y unique-TSP están en $\text{PTIME}^{\text{SAT}}$
  - También están contenidos en PTIME^A, para A problema NP-completo
- NP, co-NP y DP están contenidas en $\text{PTIME}^{\text{SAT}}$

Clases de complejidad y la noción de oráculo

Una definición mas general:

Definición

$$PTIME^{NP} = \bigcup_{A \in NP} PTIME^{A}$$

Clases de complejidad y la noción de oráculo

Una definición mas general:

Definición

$$PTIME^{NP} = \bigcup_{A \in NP} PTIME^{A}$$

En realidad esta definición no es más general:

Proposición

$PTIME^{NP} = PTIME^{SAT}$

Clases de complejidad y la noción de oráculo

Una definición mas general:

Definición

$$PTIME^{NP} = \bigcup_{A \in NP} PTIME^{A}$$

En realidad esta definición no es más general:

Proposición

$PTIME^{NP} = PTIME^{SAT}$

Ejercicio

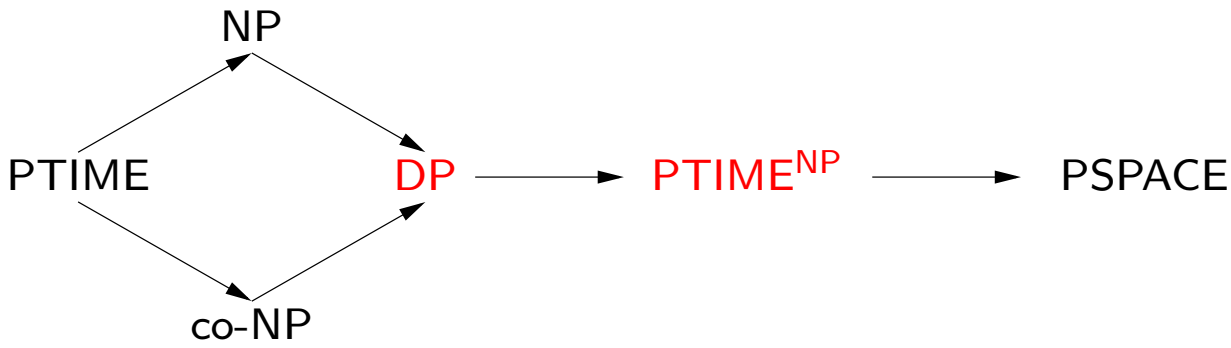
Demuestre la proposición.

¿Dónde estamos?

¿Cuál es la relación de la clase PTIME^{NP} con las clases que habíamos definido antes?

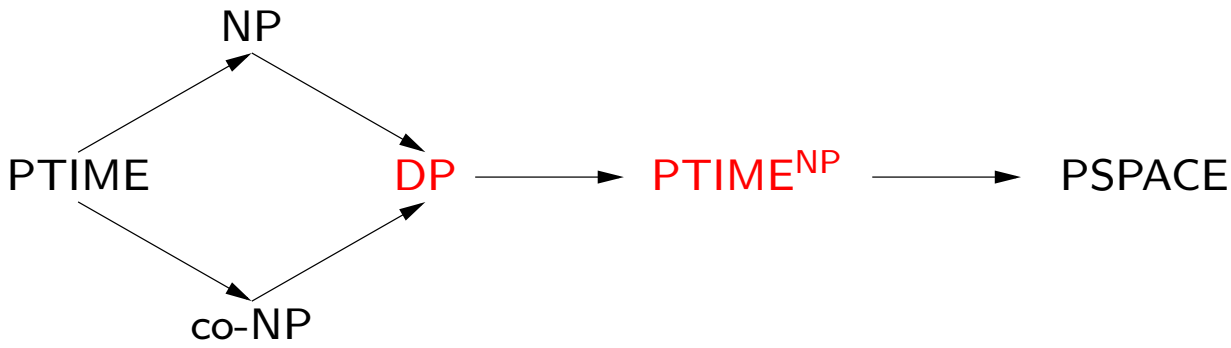
¿Dónde estamos?

¿Cuál es la relación de la clase PTIME^{NP} con las clases que habíamos definido antes?



¿Dónde estamos?

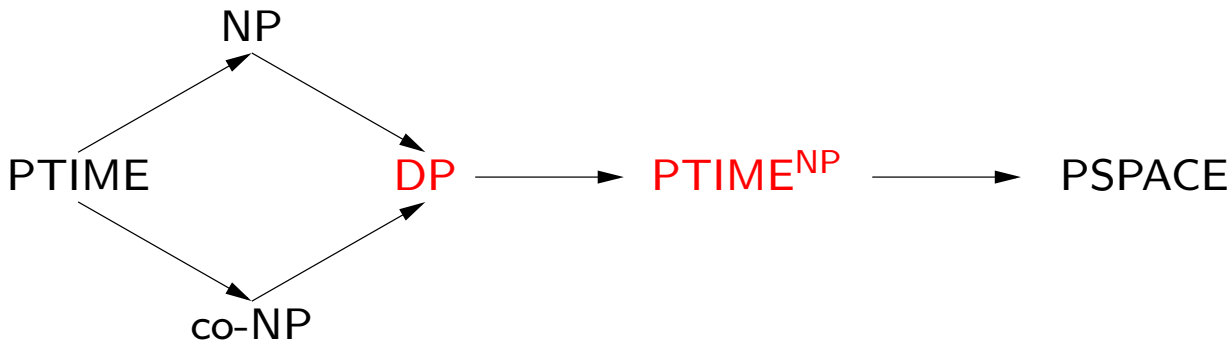
¿Cuál es la relación de la clase PTIME^{NP} con las clases que habíamos definido antes?



¿Por qué se tiene que $\text{PTIME}^{\text{NP}} \subseteq \text{PSPACE}$?

¿Dónde estamos?

¿Cuál es la relación de la clase PTIME^{NP} con las clases que habíamos definido antes?



¿Por qué se tiene que $\text{PTIME}^{\text{NP}} \subseteq \text{PSPACE}$?

¿Basta con esta clase?

- No, hay otros problemas naturales entre PTIME^{NP} y PSPACE

Un problema en inteligencia artificial: Revisión de conocimiento

Un problema fundamental: Actualización de conocimiento.

- Dada una base de conocimiento Σ y una nueva premisa φ, queremos actualizar el conocimiento en Σ de acuerdo con φ

Un problema en inteligencia artificial: Revisión de conocimiento

Un problema fundamental: Actualización de conocimiento.

- Dada una base de conocimiento Σ y una nueva premisa φ, queremos actualizar el conocimiento en Σ de acuerdo con φ
- Queremos actualizar lo que sea *necesario*

Un problema en inteligencia artificial: Revisión de conocimiento

Un problema fundamental: Actualización de conocimiento.

- Dada una base de conocimiento Σ y una nueva premisa φ, queremos actualizar el conocimiento en Σ de acuerdo con φ
- Queremos actualizar lo que sea *necesario*
- No queremos perder información que no es *necesario* eliminar

Un problema en inteligencia artificial: Revisión de conocimiento

Un problema fundamental: Actualización de conocimiento.

- Dada una base de conocimiento Σ y una nueva premisa φ, queremos actualizar el conocimiento en Σ de acuerdo con φ
- Queremos actualizar lo que sea *necesario*
- No queremos perder información que no es *necesario* eliminar

Supuestos:

- Σ es un conjunto de fórmulas proposicionales
- φ es una fórmula proposicional

Revisión de conocimiento

Dado Σ y φ: queremos generar una fórmula que refleje la actualización de Σ dado φ.

Notación

$\Sigma \circ \varphi$

Revisión de conocimiento

Dado Σ y φ: queremos generar una fórmula que refleje la actualización de Σ dado φ.

Notación

$\Sigma \circ \varphi$

¿Cómo podemos hacer esto?

- ¿Qué debería ser $\{p, p \rightarrow q\} \circ \neg q$?

Revisión de conocimiento

Una posible solución: Belief Revision

Revisión de conocimiento

Una posible solución: Belief Revision

Notación

- *modelos*(Σ)*: Conjunto de las valuaciones* σ *que satisfacen* Σ
- $\Delta(\sigma_1, \sigma_2)$*: Conjunto de las variables proposicionales* p *tales que* $\sigma_1(p) \neq \sigma_2(p)$
  - *Consideramos valuaciones con el mismo dominio*

Revisión de conocimiento

Una posible solución: Belief Revision

Notación

- *modelos*(Σ): *Conjunto de las valuaciones* σ *que satisfacen* Σ
- $\Delta(\sigma_1, \sigma_2)$: *Conjunto de las variables proposicionales* p *tales que* $\sigma_1(p) \neq \sigma_2(p)$
  - *Consideramos valuaciones con el mismo dominio*

Ejemplo

Si $\sigma_1(p) = 1$, $\sigma_1(q) = 1$, $\sigma_2(p) = 1$ y $\sigma_2(q) = 0$, entonces $\Delta(\sigma_1, \sigma_2) = \{q\}$

- $\Delta(\sigma_1, \sigma_2)$ mide la *distancia* entre σ_1 y σ_2

Revisión de conocimiento

Para actualizar Σ dado φ: Actualizamos los modelos de Σ con respecto a φ.

Dado σ tal que $\sigma(\Sigma) = 1$, queremos seleccionar los modelos σ_1 de φ que están a distancia mínima de σ.

Revisión de conocimiento

Para actualizar Σ dado φ: Actualizamos los modelos de Σ con respecto a φ.

Dado σ tal que $\sigma(\Sigma) = 1$, queremos seleccionar los modelos σ_1 de φ que están a distancia mínima de σ.

Notación

$$\textit{mínimo}(\sigma, \varphi) = \{\sigma_1 \mid \sigma_1(\varphi) = 1 \textit{ y no existe } \sigma_2 \textit{ tal que } \sigma_2(\varphi) = 1 \textit{ y } \Delta(\sigma, \sigma_2) \subsetneq \Delta(\sigma, \sigma_1)\}$$

Revisión de conocimiento

Definimos los modelos de $\Sigma \circ \varphi$ como los modelos de φ que están *más cerca* de los modelos de Σ:

$$\mathit{modelos}(\Sigma \circ \varphi) \quad = \quad \bigcup_{\sigma \,:\, \sigma(\Sigma)=1} \mathit{mínimo}(\sigma, \varphi)$$

y definimos $\Sigma \circ \varphi$ como una fórmula ψ arbitraria tal que $\mathit{modelos}(\psi) = \mathit{modelos}(\Sigma \circ \varphi)$.

Revisión de conocimiento

Definimos los modelos de $\Sigma \circ \varphi$ como los modelos de φ que están *más cerca* de los modelos de Σ:

$$\mathit{modelos}(\Sigma \circ \varphi) \quad = \quad \bigcup_{\sigma \,:\, \sigma(\Sigma)=1} \mathit{mínimo}(\sigma, \varphi)$$

y definimos $\Sigma \circ \varphi$ como una fórmula ψ arbitraria tal que $\mathit{modelos}(\psi) = \mathit{modelos}(\Sigma \circ \varphi)$.

- ¿Siempre existe esta fórmula? ¿Es única?

Revisión de conocimiento

Ejemplo

$\Sigma = \{p, p \to q\}$ y $\varphi = \neg q$

$$\begin{array}{rcll} \mathit{modelos}(\Sigma) & = & \{\sigma\}, & \sigma(p) = \sigma(q) = 1 \\ \mathit{modelos}(\varphi) & = & \{\sigma_1, \sigma_2\}, & \sigma_1(p) = 1,\ \sigma_1(q) = 0 \\ & & & \sigma_2(p) = 0,\ \sigma_2(q) = 0 \end{array}$$

Modelos mínimos:

$$\begin{array}{rcl} \Delta(\sigma, \sigma_1) & = & \{q\} \\ \Delta(\sigma, \sigma_2) & = & \{p, q\} \\ \mathit{mínimo}(\sigma, \varphi) & = & \{\sigma_1\} \\ \mathit{modelos}(\Sigma \circ \varphi) & = & \{\sigma_1\} \end{array}$$

Resultado: $\{p, p \to q\} \circ \neg q \quad = \quad p \wedge \neg q$

Revisión de conocimiento: Complejidad

Un problema fundamental en el área: Calcular respuestas *certeras*.

- ψ es una respuesta certera en $\Sigma \circ \varphi$ si para cada $\sigma \in \mathit{modelos}(\Sigma \circ \varphi)$, se tiene que σ satisface ψ

Ejemplo

p es una respuesta certera en $\{p, p \rightarrow q\} \circ \neg q$.

Revisión de conocimiento: Complejidad

Un problema fundamental en el área: Calcular respuestas *certeras*.

- ψ es una respuesta certera en $\Sigma \circ \varphi$ si para cada $\sigma \in \mathit{modelos}(\Sigma \circ \varphi)$, se tiene que σ satisface ψ

Ejemplo

p es una respuesta certera en $\{p, p \to q\} \circ \neg q$.

Problema que queremos estudiar:

$$\text{CERTAIN-ANSWERS} = \{(\Sigma, \varphi, \psi) \mid \psi \text{ es una respuesta certera en } \Sigma \circ \varphi\}$$

Revisión de conocimiento: Complejidad

¿Cuál es la complejidad de CERTAIN-ANSWERS?

Revisión de conocimiento: Complejidad

¿Cuál es la complejidad de CERTAIN-ANSWERS?

- ¿Está en NP? ¿En co-NP?

Revisión de conocimiento: Complejidad

¿Cuál es la complejidad de CERTAIN-ANSWERS?

- ¿Está en NP? ¿En co-NP?
- ¿Está en PTIME^{NP}?

Revisión de conocimiento: Complejidad

¿Cuál es la complejidad de CERTAIN-ANSWERS?

- ¿Está en NP? ¿En co-NP?
- ¿Está en PTIME^{NP}?
- ¿Al menos está en PSPACE?

Revisión de conocimiento: Complejidad

¿Cuál es la complejidad de CERTAIN-ANSWERS?

- ¿Está en NP? ¿En co-NP?
- ¿Está en PTIME^{NP}?
- ¿Al menos está en PSPACE?

Nuevamente la noción de oráculo nos puede ayudar a entender la complejidad de un problema.

Una segunda clase definida en términos de oráculos

Definición

- NP^A: *Lenguajes L para los cuales existe una MT no determinista M^A tal que $L = L(M^A)$ y M^A funciona en tiempo $O(n^k)$*
- NP^{NP}: $\bigcup_{A \in NP} NP^A$

Una segunda clase definida en términos de oráculos

Definición

- NP^A: *Lenguajes L para los cuales existe una MT no determinista M^A tal que $L = L(M^A)$ y M^A funciona en tiempo $O(n^k)$*
- NP^{NP}: $\bigcup_{A \in NP} NP^A$

Podemos describir mejor la complejidad de CERTAIN-ANSWERS.

Una segunda clase definida en términos de oráculos

Definición

- NP^A: *Lenguajes L para los cuales existe una MT no determinista M^A tal que $L = L(M^A)$ y M^A funciona en tiempo $O(n^k)$*
- NP^{NP}: $\bigcup_{A \in NP} NP^A$

Podemos describir mejor la complejidad de CERTAIN-ANSWERS.

Ejercicio

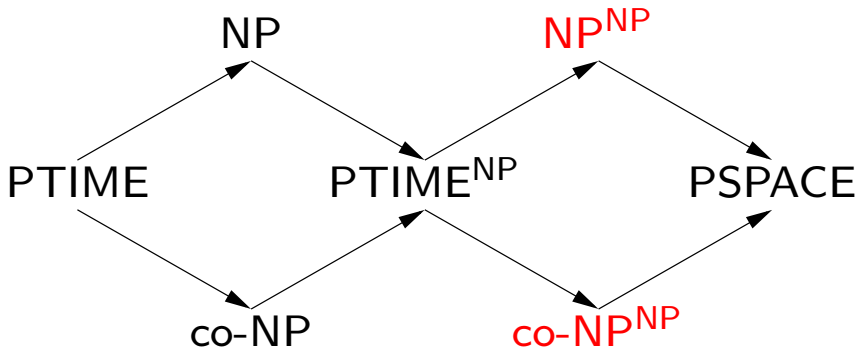
Demuestre que CERTAIN-ANSWERS $\in$ co-NP^{NP}.

¿Y ahora dónde estamos?

¿Cuál es la relación de la clase $\mathrm{NP}^{\mathrm{NP}}$ con las clases que habíamos definido antes?

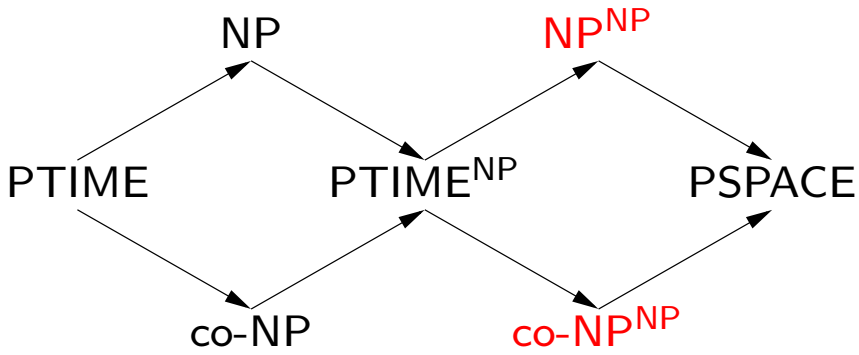
¿Y ahora dónde estamos?

¿Cuál es la relación de la clase NP^{NP} con las clases que habíamos definido antes?



¿Y ahora dónde estamos?

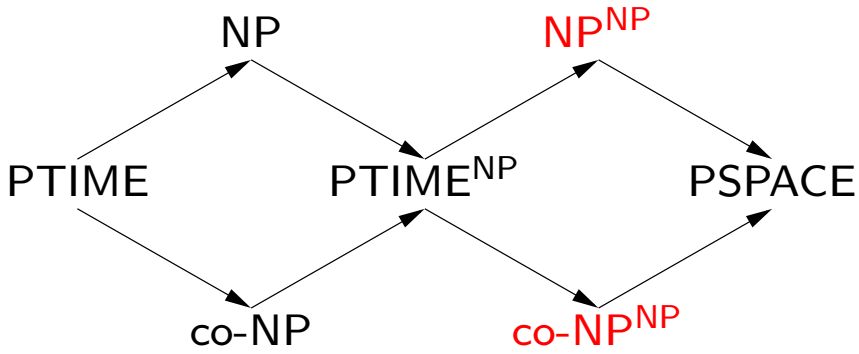
¿Cuál es la relación de la clase NP^{NP} con las clases que habíamos definido antes?



¿Por qué se tiene que $\text{NP}^{\text{NP}} \subseteq \text{PSPACE}$?

¿Y ahora dónde estamos?

¿Cuál es la relación de la clase NP^{NP} con las clases que habíamos definido antes?



¿Por qué se tiene que $\text{NP}^{\text{NP}} \subseteq \text{PSPACE}$?

Esta construcción se puede generalizar.

- Vamos a definir la jerarquía polinomial

La jerarquía polinomial

Podemos generalizar las definiciones anteriores considerando una clase de complejidad $\mathcal{C}$.

La jerarquía polinomial

Podemos generalizar las definiciones anteriores considerando una clase de complejidad $\mathcal{C}$.

Definición

- $PTIME^{\mathcal{C}}: \bigcup_{A \in \mathcal{C}} PTIME^{A}$
- $NP^{\mathcal{C}}: \bigcup_{A \in \mathcal{C}} NP^{A}$

La jerarquía polinomial

Usamos la generalización anterior para definir la jerarquía polinomial.

La jerarquía polinomial

Usamos la generalización anterior para definir la jerarquía polinomial.

Definición (Jerarquía Polinomial)

La jerarquía polinomial

Usamos la generalización anterior para definir la jerarquía polinomial.

Definición (Jerarquía Polinomial)

$$\Sigma_0^P \quad = \quad PTIME$$

La jerarquía polinomial

Usamos la generalización anterior para definir la jerarquía polinomial.

Definición (Jerarquía Polinomial)

$$\Sigma_0^P = PTIME$$

$$\Sigma_{n+1}^P = NP^{\Sigma_n^P} \qquad n \geq 0$$

La jerarquía polinomial

Usamos la generalización anterior para definir la jerarquía polinomial.

Definición (Jerarquía Polinomial)

$$\begin{aligned} \Sigma_0^P &= PTIME \\ \Sigma_{n+1}^P &= NP^{\Sigma_n^P} && n \geq 0 \\ \Delta_{n+1}^P &= PTIME^{\Sigma_n^P} && n \geq 0 \end{aligned}$$

La jerarquía polinomial

Usamos la generalización anterior para definir la jerarquía polinomial.

Definición (Jerarquía Polinomial)

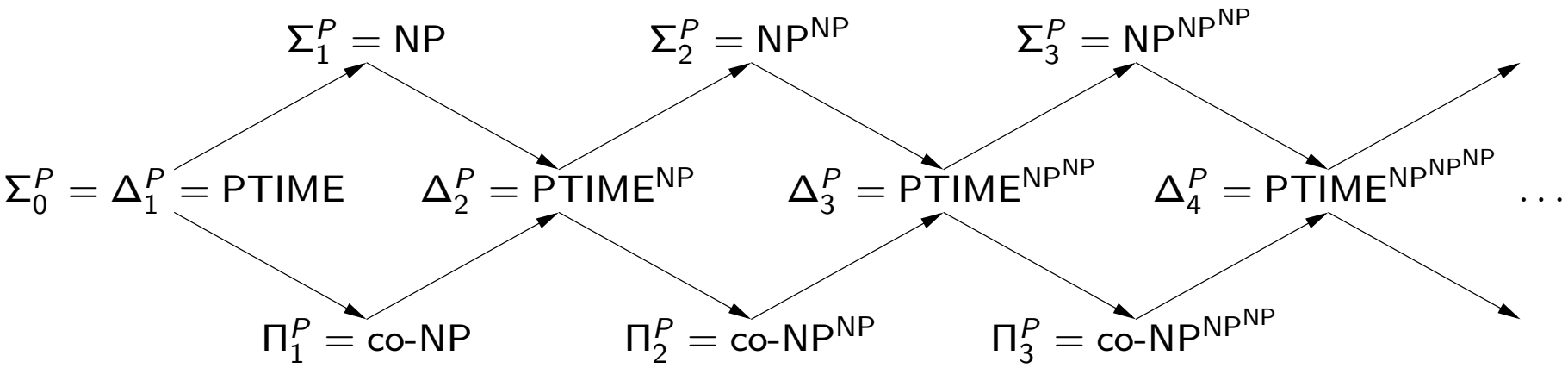
$$
\begin{aligned}
\Sigma_0^P &= PTIME \\
\Sigma_{n+1}^P &= NP^{\Sigma_n^P} && n \geq 0 \\
\Delta_{n+1}^P &= PTIME^{\Sigma_n^P} && n \geq 0 \\
\Pi_{n+1}^P &= co\text{-}\Sigma_{n+1}^P && n \geq 0
\end{aligned}
$$

La jerarquía polinomial

¿Cómo se ve la jerarquía polinomial en una figura?

La jerarquía polinomial

¿Cómo se ve la jerarquía polinomial en una figura?



La jerarquía polinomial y PSPACE

Definición

$$PH = \bigcup_{k \geq 0} \Sigma_k^P$$

La jerarquía polinomial y PSPACE

Definición

$$PH = \bigcup_{k \geq 0} \Sigma_k^P$$

¿Cuál es la relación entre PH y PSPACE?

La jerarquía polinomial y PSPACE

Definición

$$PH = \bigcup_{k \geq 0} \Sigma_k^P$$

¿Cuál es la relación entre PH y PSPACE?

- PH $\subseteq$ PSPACE

La jerarquía polinomial y PSPACE

¿Puede ser cierto que PSPACE $\subseteq$ PH?

La jerarquía polinomial y PSPACE

¿Puede ser cierto que PSPACE $\subseteq$ PH?

- Esto implicaría que PH tiene problemas completos

La jerarquía polinomial y PSPACE

¿Puede ser cierto que PSPACE $\subseteq$ PH?

- Esto implicaría que PH tiene problemas completos

¿Puede tener problemas completos PH?

La jerarquía polinomial y PSPACE

¿Puede ser cierto que PSPACE $\subseteq$ PH?

- Esto implicaría que PH tiene problemas completos

¿Puede tener problemas completos PH?

- Cada clase Σ_k^P es cerrada bajo $\leq_{\mathrm{m}}^{\log}$

La jerarquía polinomial y PSPACE

¿Puede ser cierto que PSPACE $\subseteq$ PH?

- Esto implicaría que PH tiene problemas completos

¿Puede tener problemas completos PH?

- Cada clase Σ_k^P es cerrada bajo $\leq_{\mathrm{m}}^{\log}$
- Si PH tiene un problema completo, entonces la jerarquía polinomial colapsa a algún nivel finito

La jerarquía polinomial y PSPACE

¿Puede ser cierto que PSPACE $\subseteq$ PH?

- Esto implicaría que PH tiene problemas completos

¿Puede tener problemas completos PH?

- Cada clase Σ_k^P es cerrada bajo $\leq_{\mathrm{m}}^{\log}$
- Si PH tiene un problema completo, entonces la jerarquía polinomial colapsa a algún nivel finito

Proposición

Si la jerarquía polinomial no colapsa a algún nivel finito, entonces PH $\subsetneq$ PSPACE.

La jerarquía polinomial: Un poco más de intuición

Ejercicio

Una expresión entera se define recursivamente como:

- Cada $a \in \mathbb{N}$ es una expresión entera.
- Si E y F son expresiones enteras, entonces $(E + F)$ y $(E \cup F)$ son expresiones enteras.

El conjunto de números naturales representado por una expresión entera se define recursivamente como:

- $L(a) = \{a\}$
- $L(E + F) = \{a + b \mid a \in L(E) \text{ y } b \in L(F)\}$
- $L(E \cup F) = L(E) \cup L(F)$

¿Cuál es la complejidad del problema de verificar si $L(E) = L(F)$?

La jerarquía polinomial: Una primera propiedad

Notación

$\Pi_0^P = PTIME$

Proposición

Para $k \geq 1$: Un lenguaje L sobre un alfabeto Σ está en Σ_k^P si y sólo si existe $A \in \Pi_{k-1}^P$ y un polinomio $p(n)$ tal que para todo $w \in \Sigma^*$:

$w \in L$ *si y sólo si*

existe $z \in \Sigma^*$ tal que $|z| \leq p(|w|)$ y $(w, z) \in A$.

La jerarquía polinomial: Una primera propiedad

Notación

$\Pi_0^P = PTIME$

Proposición

Para $k \geq 1$: Un lenguaje L sobre un alfabeto Σ está en Σ_k^P si y sólo si existe $A \in \Pi_{k-1}^P$ y un polinomio $p(n)$ tal que para todo $w \in \Sigma^*$:

$w \in L$ *si y sólo si*

existe $z \in \Sigma^*$ tal que $|z| \leq p(|w|)$ y $(w, z) \in A$.

Demostración: Por inducción en k

La jerarquía polinomial: Una primera propiedad

Para $k = 1$: Tenemos que demostrar que $L \in \text{NP}$ si y sólo si existe un lenguaje $A \in \text{PTIME}$ y un polinomio $p(n)$ tal que:

$$w \in L \quad \text{si y sólo si} \quad \text{existe } z \text{ tal que } |z| \leq p(|w|) \text{ y } (w, z) \in A.$$

La jerarquía polinomial: Una primera propiedad

Para $k = 1$: Tenemos que demostrar que $L \in \text{NP}$ si y sólo si existe un lenguaje $A \in \text{PTIME}$ y un polinomio $p(n)$ tal que:

$$w \in L \quad \text{si y sólo si} \quad \text{existe } z \text{ tal que } |z| \leq p(|w|) \text{ y } (w, z) \in A.$$

¿Cómo se demuestra esto?

- La dirección $\Leftarrow$ es simple
- ¿Cómo se demuestra la otra dirección? ¿Qué lenguaje A hay que usar?

La jerarquía polinomial: Una primera propiedad

Suponga que la propiedad es cierta para k. Vamos a demostrar que también se cumple para $k+1$.

($\Leftarrow$) Suponga que existe un lenguaje $A \in \Pi_k^P$ y un polinomio $p(n)$ tal que:

$$w \in L \quad \text{si y sólo si} \quad \text{existe } z \text{ tal que } |z| \leq p(|w|) \text{ y } (w, z) \in A.$$

La jerarquía polinomial: Una primera propiedad

Suponga que la propiedad es cierta para k. Vamos a demostrar que también se cumple para $k+1$.

($\Leftarrow$) Suponga que existe un lenguaje $A \in \Pi_k^P$ y un polinomio $p(n)$ tal que:

$w \in L$ si y sólo si existe z tal que $|z| \leq p(|w|)$ y $(w, z) \in A$.

Entonces se tiene que $L \in \mathrm{NP}^A$

- Por lo tanto: $L \in \Sigma_{k+1}^P$

La jerarquía polinomial: Una primera propiedad

($\Rightarrow$) Suponga que $L \in \Sigma_{k+1}^P$.

Se tiene que $L = L(M^A)$, donde

- $A \in \Sigma_k^P$
- M^A es una MT no determinista que funciona en tiempo polinomial y tiene un oráculo para A

Dado que $A \in \Sigma_k^P$, tenemos por hipótesis de inducción que existe $C \in \Pi_{k-1}^P$ y un polinomio $p(n)$ tal que:

$w \in A$ si y sólo si existe z tal que $|z| \leq p(|w|)$ y $(w, z) \in C$.

La jerarquía polinomial: Una primera propiedad

Sea Γ el alfabeto de la cinta de M^A y suponga que $\# \notin \Gamma$.

Definimos el lenguaje D como el conjunto de los pares (w, z) tales que $w \in \Sigma^*$, $z = \alpha_0 \# \beta_0 \# \cdots \# \alpha_{m-1} \# \beta_{m-1} \# \alpha_m$, y:

(a) α_0 es la configuración inicial de M^A con entrada w

(b) α_m es una configuración de aceptación de M^A

(c) para $i \in [0, m-1]$: Si el estado de α_i no es $q_?$, entonces α_{i+1} es una configuración sucesora de α_i en M^A y $\beta_i = \varepsilon$

La jerarquía polinomial: Una primera propiedad

(d) para $i \in [0, m-1]$: Si el estado de α_i es $q_?$ y $u \in \Sigma^*$ es la palabra en la cinta de consulta en α_i, entonces α_{i+1} sólo difiere de α_i en el estado y:

(d.1) el estado de α_{i+1} es q_{NO}, $u \notin A$ y $\beta_i = \varepsilon$; ó

(d.2) el estado de α_{i+1} es q_{YES}, $|\beta_i| \leq p(|u|)$ y $(u, \beta_i) \in C$

La jerarquía polinomial: Una primera propiedad

(d) para $i \in [0, m-1]$: Si el estado de α_i es $q_?$ y $u \in \Sigma^*$ es la palabra en la cinta de consulta en α_i, entonces α_{i+1} sólo difiere de α_i en el estado y:

(d.1) el estado de α_{i+1} es q_{NO}, $u \notin A$ y $\beta_i = \varepsilon$; ó

(d.2) el estado de α_{i+1} es q_{YES}, $|\beta_i| \leq p(|u|)$ y $(u, \beta_i) \in C$

Lema

Existe un polinomio $q(n)$ tal que para cada $w \in \Sigma^*$:

$w \in L$ *si y sólo si*

existe $z \in \Sigma^*$ tal que $|z| \leq q(|w|)$ y $(w, z) \in D$.

La jerarquía polinomial: Una primera propiedad

Vamos a demostrar que $D \in \Pi_k^P$.

Sea M^C una MT con oráculo para C que con entrada (w, z) funciona de la siguiente manera:

(1) Verifica en tiempo polinomial si (w, z) no cumple (a), (b) ó (c), y si es así M^C acepta (w, z)

(2) Si no se cumple (1), entonces M^C hace lo siguiente para cada configuración α_i con estado $q_?$:

  (2.1) Si el estado de α_{i+1} no es ni q_{NO} ni q_{YES}, si α_i y α_{i+1} difieren en algún símbolo que no corresponde al estado, o si el estado de α_{i+1} es q_{NO} y $\beta_i \neq \varepsilon$, entonces M^C acepta (w, z)

La jerarquía polinomial: Una primera propiedad

(2.2) Si no se cumple (2.1), el estado de α_{i+1} es q_{NO} y $u \in \Sigma^*$ es la palabra en la cinta de consulta en α_i, entonces M^C adivina v tal que $|v| \leq p(|u|)$, y después invoca al oráculo C con entrada (u, v). Si el oráculo responde sí, entonces M^C acepta (w, z)

(2.3) Si no se cumplen (2.1) y (2.2), el estado de α_{i+1} es q_{YES}, $u \in \Sigma^*$ es la palabra en la cinta de consulta en α_i y $|\beta_i| > p(|u|)$, entonces M^C acepta (w, z)

(2.4) Si no se cumplen (2.1), (2.2) y (2.3), el estado de α_{i+1} es q_{YES} y $u \in \Sigma^*$ es la palabra en la cinta de consulta en α_i, entonces M^C invoca al oráculo C con entrada (u, β_i). Si el oráculo responde no, entonces M^C acepta (w, z)

La jerarquía polinomial: Una primera propiedad

(2.2) Si no se cumple (2.1), el estado de α_{i+1} es q_{NO} y $u \in \Sigma^*$ es la palabra en la cinta de consulta en α_i, entonces M^C adivina v tal que $|v| \leq p(|u|)$, y después invoca al oráculo C con entrada (u, v). Si el oráculo responde sí, entonces M^C acepta (w, z)

(2.3) Si no se cumplen (2.1) y (2.2), el estado de α_{i+1} es q_{YES}, $u \in \Sigma^*$ es la palabra en la cinta de consulta en α_i y $|\beta_i| > p(|u|)$, entonces M^C acepta (w, z)

(2.4) Si no se cumplen (2.1), (2.2) y (2.3), el estado de α_{i+1} es q_{YES} y $u \in \Sigma^*$ es la palabra en la cinta de consulta en α_i, entonces M^C invoca al oráculo C con entrada (u, β_i). Si el oráculo responde no, entonces M^C acepta (w, z)

No es difícil demostrar que $\overline{D} = L(M^C)$.

La jerarquía polinomial: Una primera propiedad

(2.2) Si no se cumple (2.1), el estado de α_{i+1} es q_{NO} y $u \in \Sigma^*$ es la palabra en la cinta de consulta en α_i, entonces M^C adivina v tal que $|v| \leq p(|u|)$, y después invoca al oráculo C con entrada (u, v). Si el oráculo responde sí, entonces M^C acepta (w, z)

(2.3) Si no se cumplen (2.1) y (2.2), el estado de α_{i+1} es q_{YES}, $u \in \Sigma^*$ es la palabra en la cinta de consulta en α_i y $|\beta_i| > p(|u|)$, entonces M^C acepta (w, z)

(2.4) Si no se cumplen (2.1), (2.2) y (2.3), el estado de α_{i+1} es q_{YES} y $u \in \Sigma^*$ es la palabra en la cinta de consulta en α_i, entonces M^C invoca al oráculo C con entrada (u, β_i). Si el oráculo responde no, entonces M^C acepta (w, z)

No es difícil demostrar que $\overline{D} = L(M^C)$.

- ¿Cómo se demuestra esto?

La jerarquía polinomial: Una primera propiedad

Concluimos que $D \in \Pi_k^P$.

- Ya que M^C es una MT no determinista que funciona en tiempo polinomial y tiene un oráculo para $C \in \Pi_{k-1}^P$

La jerarquía polinomial: Una primera propiedad

Concluimos que $D \in \Pi_k^P$.

- Ya que M^C es una MT no determinista que funciona en tiempo polinomial y tiene un oráculo para $C \in \Pi_{k-1}^P$

Para terminar sólo nos queda considerar el alfabeto de D.

- La entrada de D es de la forma (w, z), donde $z \in (\Gamma \cup \{\#\})^*$

La jerarquía polinomial: Una primera propiedad

Concluimos que $D \in \Pi_k^P$.

- Ya que M^C es una MT no determinista que funciona en tiempo polinomial y tiene un oráculo para $C \in \Pi_{k-1}^P$

Para terminar sólo nos queda considerar el alfabeto de D.

- La entrada de D es de la forma (w, z), donde $z \in (\Gamma \cup \{\#\})^*$

Pero: El uso del alfabeto $\Gamma \cup \{\#\}$ no es es esencial para la definición de D.

- Podemos suponer que $z \in \Sigma^*$. ¿Por qué?

La jerarquía polinomial: Caracterizando Σ_k^P

Teorema

Para $k \geq 1$: Un lenguaje L sobre un alfabeto Σ está en Σ_k^P si y sólo si existe $A \in PTIME$ y un polinomio $p(n)$ tal que para todo $w \in \Sigma^*$:

$$w \in L \text{ si y sólo si}$$
$$(\exists z_1 \in \Sigma^*,\ |z_1| \leq p(|w|))\ (\forall z_2 \in \Sigma^*,\ |z_2| \leq p(|w|))\ \cdots$$
$$(Q_k z_k \in \Sigma^*,\ |z_k| \leq p(|w|))\ (w, z_1, z_2, \ldots, z_k) \in A,$$

donde $Q_k = \exists$ si k es impar y $Q_k = \forall$ si k es par.

La jerarquía polinomial: Caracterizando Σ_k^P

Teorema

Para $k \geq 1$: Un lenguaje L sobre un alfabeto Σ está en Σ_k^P si y sólo si existe $A \in PTIME$ y un polinomio $p(n)$ tal que para todo $w \in \Sigma^*$:

$$w \in L \text{ si y sólo si}$$
$$(\exists z_1 \in \Sigma^*,\ |z_1| \leq p(|w|))\ (\forall z_2 \in \Sigma^*,\ |z_2| \leq p(|w|))\ \cdots$$
$$(Q_k z_k \in \Sigma^*,\ |z_k| \leq p(|w|))\ (w, z_1, z_2, \ldots, z_k) \in A,$$

donde $Q_k = \exists$ si k es impar y $Q_k = \forall$ si k es par.

Ejercicio

Demuestre el teorema.

La jerarquía polinomial: Caracterizando Σ_k^P

La caracterización anterior nos va a servir para encontrar problemas completos para cada clase Σ_k^P.

- Cada uno de estos problemas es una extensión de SAT

La jerarquía polinomial: Caracterizando Σ_k^P

La caracterización anterior nos va a servir para encontrar problemas completos para cada clase Σ_k^P.

- Cada uno de estos problemas es una extensión de SAT

Pero antes vamos a demostrar un resultado fundamental sobre la jerarquía polinomial.

- Vamos a razonar sobre un posible colapso de la jerarquía polinomial
- En la demostración, suponemos la existencia de problemas completos para cada clase Σ_k^P

El colapso de la jerarquía polinomial

Teorema

Para $k \geq 1$:

(a) *Si $\Sigma_k^P = \Pi_k^P$, entonces $PH = \Sigma_k^P$*

(b) *Si $\Sigma_k^P = \Delta_k^P$, entonces $PH = \Delta_k^P$*

El colapso de la jerarquía polinomial

Teorema

Para $k \geq 1$*:*

(a) *Si* $\Sigma_k^P = \Pi_k^P$, *entonces* $PH = \Sigma_k^P$

(b) *Si* $\Sigma_k^P = \Delta_k^P$, *entonces* $PH = \Delta_k^P$

Demostración: Usamos el siguiente lema:

Lema

Para $k \geq 1$: $NP^{\Sigma_k^P \cap \Pi_k^P} = \Sigma_k^P$

Demostración del lema:

($\supseteq$) Si $L \in \Sigma_k^P$, entonces $L \in \mathrm{NP}^A$, para $A \in \Sigma_{k-1}^P$.

Pero $\Sigma_{k-1}^P \subseteq \Delta_k^P \subseteq \Sigma_k^P \cap \Pi_k^P$.

- Concluimos que $L \in \mathrm{NP}^{\Sigma_k^P \cap \Pi_k^P}$

($\subseteq$) Suponga que $L \in \mathrm{NP}^{\Sigma_k^P \cap \Pi_k^P}$.

Se tiene que $L = \mathrm{NP}^A$, para $A \in \Sigma_k^P \cap \Pi_k^P$.

El colapso de la jerarquía polinomial

Como $A \in \Sigma_k^P \cap \Pi_k^P$: $A = L(M_1^B)$ y $\overline{A} = L(M_2^B)$, donde:

- B es un problema completo para Σ_{k-1}^P
- M_1^B y M_2^B son MTs no deterministas que funcionan en tiempo polinomial y tienen oráculo para B

El colapso de la jerarquía polinomial

Como $A \in \Sigma_k^P \cap \Pi_k^P$: $A = L(M_1^B)$ y $\overline{A} = L(M_2^B)$, donde:

- B es un problema completo para Σ_{k-1}^P
- M_1^B y M_2^B son MTs no deterministas que funcionan en tiempo polinomial y tienen oráculo para B

Por lo tanto: $L \in \mathrm{NP}^B$

El colapso de la jerarquía polinomial

Como $A \in \Sigma_k^P \cap \Pi_k^P$: $A = L(M_1^B)$ y $\overline{A} = L(M_2^B)$, donde:

- B es un problema completo para Σ_{k-1}^P
- M_1^B y M_2^B son MTs no deterministas que funcionan en tiempo polinomial y tienen oráculo para B

Por lo tanto: $L \in \mathrm{NP}^B$

- Idea: Reemplazamos cada llamada en M^A al oráculo A por una llamada simultanea a las rutinas M_1^B y M_2^B

El colapso de la jerarquía polinomial

Como $A \in \Sigma_k^P \cap \Pi_k^P$: $A = L(M_1^B)$ y $\overline{A} = L(M_2^B)$, donde:

- B es un problema completo para Σ_{k-1}^P
- M_1^B y M_2^B son MTs no deterministas que funcionan en tiempo polinomial y tienen oráculo para B

Por lo tanto: $L \in \text{NP}^B$

- Idea: Reemplazamos cada llamada en M^A al oráculo A por una llamada simultanea a las rutinas M_1^B y M_2^B

Concluimos que $L \in \Sigma_k^P$. □

El colapso de la jerarquía polinomial

Ahora volvemos a la demostración inicial.

(a) Suponemos que $\Sigma_k^P = \Pi_k^P$.

Vamos a demostrar por inducción en $j \geq k$ que $\Sigma_j^P = \Pi_j^P = \Sigma_k^P$.

Para $j = k$ la propiedad se tiene por hipótesis.

Suponemos que la propiedad se cumple para $j \geq k$, y vamos a demostrar que también es cierta para $j + 1$.

El colapso de la jerarquía polinomial

Tenemos que:

$$
\begin{array}{lcll}
\Sigma_{j+1}^P & = & \mathrm{NP}^{\Sigma_j^P} & \\
& = & \mathrm{NP}^{\Sigma_j^P \cap \Pi_j^P} & \text{ya que } \Sigma_j^P = \Pi_j^P \\
& = & \Sigma_j^P & \text{por lema } (j \geq k \geq 1) \\
& = & \Sigma_k^P & \text{por hipótesis de inducción}
\end{array}
$$

Además tenemos que:

$$
\begin{array}{lcll}
\Pi_{j+1}^P & = & \text{co-}\Sigma_{j+1}^P & \\
& = & \text{co-}\Sigma_k^P & \text{por demostración de arriba} \\
& = & \Sigma_k^P & \text{por hipótesis}
\end{array}
$$

El colapso de la jerarquía polinomial

(b) Suponemos que $\Delta_k^P = \Sigma_k^P$.

Como Δ_k^P es cerrada bajo complemento, tenemos que $\Sigma_k^P = \Pi_k^P$.

- Por (a) tenemos que $\text{PH} = \Sigma_k^P$

Concluimos que $\text{PH} = \Delta_k^P$. □

Problemas completos en la jerarquía polinomial

El lenguaje QBF_i ($i \geq 1$) está formado por todas las fórmulas proposicionales cuantificadas que son válidas y de la forma:

$$\begin{aligned} &\exists x_{1,1} \cdots \exists x_{1,m_1} \\ &\forall x_{2,1} \cdots \forall x_{2,m_2} \\ &\exists x_{3,1} \cdots \exists x_{3,m_3} \\ &\cdots \\ &Q_i x_{i,1} \cdots Q_i x_{i,m_i}\ \varphi \end{aligned}$$

donde:

- $Q_i = \exists$ si i es impar y $Q_i = \forall$ si i es par
- φ es una fórmula proposicional sobre las variables $x_{1,1}, \ldots, x_{1,m_1}, \ldots, x_{i,1}, \ldots, x_{i,m_i}$

Un problema completo para Σ_k^P

La clase de problemas $\{\text{QBF}_i\}_{i \geq 1}$ es adecuada para representar la jerarquía polinomial.

Teorema

Para cada $k \geq 1$, QBF_k es Σ_k^P-completo.

Un problema completo para Σ_k^P

La clase de problemas $\{\text{QBF}_i\}_{i \geq 1}$ es adecuada para representar la jerarquía polinomial.

Teorema

Para cada $k \geq 1$, QBF_k es Σ_k^P-completo.

Demostración: Primero tenemos que demostrar que $\text{QBF}_k \in \Sigma_k^P$.

- ¿Cómo se demuestra esto?

Un problema completo para Σ_k^P: Demostración

En segundo lugar, tenemos que demostrar que QBF_k es Σ_k^P-hard.

- Sea $L \in \Sigma_k^P$. Tenemos que demostrar que L es reducible en espacio logarítmico a QBF_k

Un problema completo para Σ_k^P: Demostración

En segundo lugar, tenemos que demostrar que QBF_k es Σ_k^P-hard.

- Sea $L \in \Sigma_k^P$. Tenemos que demostrar que L es reducible en espacio logarítmico a QBF_k

Para simplificar la demostración, suponemos que:

Un problema completo para Σ_k^P: Demostración

En segundo lugar, tenemos que demostrar que QBF_k es Σ_k^P-hard.

- Sea $L \in \Sigma_k^P$. Tenemos que demostrar que L es reducible en espacio logarítmico a QBF_k

Para simplificar la demostración, suponemos que:

- L es está definido sobre el alfabeto $\{0, 1\}$

Un problema completo para Σ_k^P: Demostración

En segundo lugar, tenemos que demostrar que QBF_k es Σ_k^P-hard.

- Sea $L \in \Sigma_k^P$. Tenemos que demostrar que L es reducible en espacio logarítmico a QBF_k

Para simplificar la demostración, suponemos que:

- L es está definido sobre el alfabeto $\{0, 1\}$
- k es impar

Un problema completo para Σ_k^P: Demostración

En segundo lugar, tenemos que demostrar que QBF_k es Σ_k^P-hard.

- Sea $L \in \Sigma_k^P$. Tenemos que demostrar que L es reducible en espacio logarítmico a QBF_k

Para simplificar la demostración, suponemos que:

- L es está definido sobre el alfabeto $\{0, 1\}$
- k es impar

La demostración es similar para un alfabeto arbitrario y/o k par.

Un problema completo para Σ_k^P: Demostración

Por demostrar: Existe una función $f : \{0,1\}^* \to \{0,1\}^*$ tal que

- f es computable en espacio logarítmico
- para cada $w \in \{0,1\}^*$: $w \in L$ si y sólo si $f(w) \in \text{QBF}_k$

Notación

$f(w) = \varphi_w$

Un problema completo para Σ_k^P: Demostración

Como $L \in \Sigma_k^P$, sabemos que existe $A \in$ PTIME y un polinomio $p(n)$ tal que para todo $w \in \{0,1\}^*$:

$w \in L$ si y sólo si

$(\exists z_1 \in \{0,1\}^*, |z_1| = p(|w|))$

$(\forall z_2 \in \{0,1\}^*, |z_2| = p(|w|))$

$\ldots$

$(\exists z_k \in \{0,1\}^*, |z_k| = p(|w|))$ $w\#z_1\#z_2\#\ldots\#z_k \in A$

Esto se deriva de las caracterizaciones que habíamos visto antes.

- ¿Cómo se deduce que $|z_1| = |z_2| = \cdots = |z_k| = p(|w|)$?

Un problema completo para Σ_k^P: Demostración

Como $A \in$ PTIME, existe una MT determinista $M = (Q = \{q_0, \ldots, q_m\}, \Sigma = \{0, 1, \#\}, \Gamma = \{0, 1, \#, \mathtt{B}, \vdash\}, q_0, \delta, F)$ tal que:

- M tiene una cinta de lectura/trabajo
- M para en todas las entradas
- $L = L(M)$
- $t_M(n)$ es $O(n^c)$, donde $c > 0$ es un número natural

Un problema completo para Σ_k^P: Demostración

Sin perdida de generalidad suponemos que:

- $w = a_1 \cdots a_n$, donde cada $a_i \in \{0, 1\}$
- $F = \{q_m\}$
- Para cada $a \in \Gamma$, se tiene que $\delta(q_m, a)$ no está definido

¿Por qué podemos suponer esto?

Un problema completo para Σ_k^P: Demostración

Sea $\ell = n + k \cdot (1 + p(n))$

- ℓ es el largo de la entrada $w\#z_1\#z_2\#\cdots\#z_k$ de la MT M

Un problema completo para Σ_k^P: Demostración

Sea $\ell = n + k \cdot (1 + p(n))$

- ℓ es el largo de la entrada $w\#z_1\#z_2\#\cdots\#z_k$ de la MT M

Usamos las siguientes variables proposicionales para definir φ_w:

$$
\begin{array}{lcl}
z_{i,j} & : & i \in [1,k] \text{ y } j \in [1,p(n)] \\
s_{t,p,a} & : & t \in [0,t_M(\ell)],\ p \in [0,t_M(\ell)+1] \text{ y } a \in \{0,1,\#,\mathtt{B},\vdash\} \\
c_{t,p} & : & t \in [0,t_M(\ell)] \text{ y } p \in [0,t_M(\ell)+1] \\
e_{t,q} & : & t \in [0,t_M(\ell)] \text{ y } q \in Q
\end{array}
$$

Un problema completo para Σ_k^P: Demostración

La fórmula φ_w se define como:

$$\exists z_{1,1} \cdots \exists z_{1,p(n)}$$

$$\forall z_{2,1} \cdots \forall z_{2,p(n)}$$

$$\exists z_{3,1} \cdots \exists z_{3,p(n)}$$

$$\cdots$$

$$\exists z_{k,1} \cdots \exists z_{k,p(n)}$$

$$\exists s_{0,0,0}\, \exists s_{0,0,1}\, \exists s_{0,0,\#}\, \exists s_{0,0,\mathtt{B}}\, \exists s_{0,0,\vdash} \cdots \exists s_{t_M(\ell),t_M(\ell)+1,0}$$

$$\exists s_{t_M(\ell),t_M(\ell)+1,1}\, \exists s_{t_M(\ell),t_M(\ell)+1,\#}\, \exists s_{t_M(\ell),t_M(\ell)+1,\mathtt{B}}\, \exists s_{t_M(\ell),t_M(\ell)+1,\vdash}$$

$$\exists c_{0,0} \cdots \exists c_{t_M(\ell),t_M(\ell)+1}\, \exists e_{0,q_0} \cdots \exists e_{0,q_m}\, \exists e_{t_M(\ell),q_0} \cdots \exists e_{t_M(\ell),q_m}$$

$$\Big(\varphi_I \wedge \varphi_C \wedge \varphi_\delta \wedge \varphi_A \Big)$$

Un problema completo para Σ_k^P: Demostración

φ_I: Estado inicial

$$c_{0,1} \wedge e_{0,q_0} \wedge s_{0,0,\vdash} \wedge \left(\bigwedge_{p=1}^{n} s_{0,p,a_p} \right) \wedge$$

$$\left(\bigwedge_{i=1}^{k} s_{0,(n+1)+(i-1)\cdot(p(n)+1),\#} \right) \wedge$$

$$\left(\bigwedge_{i=1}^{k} \bigwedge_{j=1}^{p(n)} \neg z_{i,j} \rightarrow s_{0,(n+1)+(i-1)\cdot(p(n)+1)+j,0} \right) \wedge$$

$$\left(\bigwedge_{i=1}^{k} \bigwedge_{j=1}^{p(n)} z_{i,j} \rightarrow s_{0,(n+1)+(i-1)\cdot(p(n)+1)+j,1} \right) \wedge$$

$$\left(\bigwedge_{p=(n+1)+k\cdot(p(n)+1)}^{t_M(\ell)+1} s_{0,p,\mathtt{B}} \right)$$

Un problema completo para Σ_k^P: Demostración

φ_C: La máquina funciona correctamente

φ_C se define como la conjunción de cuatro fórmulas. Primero, cada celda siempre contiene un único símbolo:

$$\bigwedge_{t=0}^{t_M(\ell)} \bigwedge_{p=0}^{t_M(\ell)+1} \left(\bigvee_{a \in \Gamma} \left(s_{t,p,a} \wedge \bigwedge_{b \in (\Gamma \setminus \{a\})} \neg s_{t,p,b} \right) \right)$$

Un problema completo para Σ_k^P: Demostración

Segundo, la máquina siempre está en un único estado:

$$\bigwedge_{t=0}^{t_M(\ell)} \left(\bigvee_{q \in Q} \left(e_{t,q} \wedge \bigwedge_{q' \in (Q \setminus \{q\})} \neg e_{t,q'} \right) \right)$$

Tercero, la cabeza lectora siempre está en una única posición:

$$\bigwedge_{t=0}^{t_M(\ell)} \left(\bigvee_{p=0}^{t_M(\ell)+1} \left(c_{t,p} \wedge \bigwedge_{p' \in ([0, t_M(\ell)+1] \setminus \{p\})} \neg c_{t,p'} \right) \right)$$

Un problema completo para Σ_k^P: Demostración

Cuarto, el valor de una celda de una cinta de trabajo no cambia si no es apuntada por la cabeza lectora:

$$\bigwedge_{t=0}^{t_M(\ell)-1} \bigwedge_{p=0}^{t_M(\ell)+1} \left((\neg C_{t,p}) \rightarrow \bigwedge_{a \in \Gamma} (S_{t+1,p,a} \leftrightarrow S_{t,p,a}) \right)$$

Un problema completo para Σ_k^P: Demostración

φ_δ: relación δ define como funciona la máquina (representamos $\leftarrow$ como -1, $\square$ como 0 y $\rightarrow$ como 1)

$$\bigwedge_{t=0}^{t_M(\ell)-1} \bigwedge_{p=0}^{t_M(\ell)} \left[\bigwedge_{\delta(q,a)=(q',b,X)} \Big((e_{t,q} \wedge c_{t,p} \wedge s_{t,p,a}) \rightarrow (e_{t+1,q'} \wedge c_{t+1,p+X} \wedge s_{t+1,p,b}) \Big) \right] \wedge$$

$$\bigwedge_{t=0}^{t_M(\ell)-1} \bigwedge_{p=0}^{t_M(\ell)} \left[\bigwedge_{\delta(q,a) \text{ no está definido}} \Big((e_{t,q} \wedge c_{t,p} \wedge s_{t,p,a}) \rightarrow (e_{t+1,q} \wedge c_{t+1,p} \wedge s_{t+1,p,a}) \Big) \right]$$

Un problema completo para Σ_k^P: Demostración

φ_A: La máquina acepta w

$$e_{t_M(\ell), q_m}$$

Un problema completo para Σ_k^P: Demostración

φ_A: La máquina acepta w

$$e_{t_M(\ell), q_m}$$

Para terminar la demostración tenemos que mostrar que:

Un problema completo para Σ_k^P: Demostración

φ_A: La máquina acepta w

$$e_{t_M(\ell),q_m}$$

Para terminar la demostración tenemos que mostrar que:

- $w \in L$ si y sólo si φ_w es válida

Un problema completo para Σ_k^P: Demostración

φ_A: La máquina acepta w

$$e_{t_M(\ell), q_m}$$

Para terminar la demostración tenemos que mostrar que:

- $w \in L$ si y sólo si φ_w es válida
- φ_w puede ser construida en espacio logarítmico

Un problema completo para Σ_k^P: Demostración

φ_A: La máquina acepta w

$$e_{t_M(\ell), q_m}$$

Para terminar la demostración tenemos que mostrar que:

- $w \in L$ si y sólo si φ_w es válida
- φ_w puede ser construida en espacio logarítmico

¿Cómo se hace esto?